2. PRIMER

INTRODUCTION

This section of the UNIX System User's Guide provides the information that users will need to know to access the UNIX operating system. It is not intended to be a detailed description. Many of the subjects described are discussed in detail in other sections of this volume or the UNIX System User's Manual.

Throughout this section, each reference of the form **name**(1M), **name**(7), or **name**(8) refers to entries in the UNIX System Administrator's Manual. All other references to entries of the form **name**(N), where "N" is a number (1 through 6) possibly followed by a letter, refer to entry **name** in section **N** of the UNIX System User's Manual.

In this primer, software programs that can be executed by users are referred to as **programs**. A program that is in some state of execution is referred to as a **process**. The request typed by the user is referred to as a **command** or "**command line**."

In this primer, the following graphic conventions are used in examples:

RETURN	Indicates that the user should press the RETURN key on the terminal keyboard.
DEL	Indicates that the user should press the key marked DEL, DELETE, or RUBOUT (whichever is appropriate for the terminal being used).

HUMAN INTERFACE

A. Concept of a Login

The UNIX operating system is accessed by the use of a ***login***. A login is used by the system to uniquely identify users. Before the user can access the system, the user must be assigned a login by the system administrator. Every login consists of the following components:

login name
user identification number (uid)
group identification number (gid)
password.

A ***login name*** is a unique string of letters (should be all lowercase) and/or numbers that identifies an individual to the system. The login name must begin with a letter. In many cases, a person's login name is their real first name, last name, initials, or nickname. Any string of letters and/or digits can be used as your login name, as long as it is ***unique*** (i.e., different from all other login names). Only the first eight characters of a login name are used by the system. Login names are assigned by the system administrator.

The ***uid*** of a login is a unique number assigned to each login by the system administrator. This number is used by the system to identify the owners of information stored on the system and the commands that users are executing.

The ***gid*** is a unique number assigned by the system administrator to each group. This number identifies ***groups*** of users that have something in common. For example, all logins used by people in the same department (or working on the same project) may have the same gid. The gid is important for security and accounting reasons. The impact of gid numbers on the user and the group that the user belongs to is described later.

The ***password*** is a string of letters, numbers, and/or punctuation that serves to control access to a login. The password for a login is the main security feature of the UNIX operating system. Usually, every login is assigned a password. When a user ***logs in*** to the system, the password (if any) assigned to the login being used

is requested. Access to the system is not permitted until the correct password is entered. The user can change a password as needed to ensure that others are not accessing the user's login (and consequently the user's data). Any string of letters, numbers, etc., can be used as a password as long as it is more than five characters in length and composed of uppercase letters, lowercase letters, numbers, or punctuation.

It is recommended that obvious strings such as the user's social security number, birth date, or other data that could be well known about the user not be used as passwords. If the password is something that is well known about the user, someone could gain access to the user's login with little effort. The more unusual your password, the more effective your security.

B. Logging In

In order to log in, the power to the terminal must be turned on and the appropriate switches set. Depending on the type of terminal and communication link, the user may need to press the return or break key a couple of times. This is to synchronize your terminal with the system. When communication is established, the system will prompt with:

login:

The user should type in his/her login name followed by a return. After the system digests your login name, it will prompt for your password with:

Password:

The user should then type his/her password followed by a return. The system does not echo your password on the terminal as you type it in. This is an extra security measure. If you entered your login name and password correctly, the system may print one or more "messages of the day". Following the messages, the system will prompt you with the primary prompt string, which is usually the $ symbol. If a mistake is made while logging in or the system adminstrator has not set up the user's login on the system yet, the following error message is printed:

login incorrect

This error message is followed by the login: message. The user should attempt to login again.

The UNIX operating system has a hierarchy of *directories*. When the system administrator gave the user a login name, the administrator also created a "directory" for the user. This directory ordinarily is the same name as the user login name and is known as the *login* or *home* directory of the user. When the user logs in, the *home* directory becomes the current directory or working directory of the user. Any file created under the login name (assuming no other subdirectories have been created yet) is by default in the home directory. The user may, however, create one or more directories under the *home* directory. The user may then change to subdirectories by appropriate use of a "change directory" command. See **cd**(1) for details. Under a directory or a subdirectory, the user may create files as necessary. The user is the owner of the *home* directory and all subdirectories created under the *home* directory. As the owner, the user has full permission to create, alter, and remove (destroy) all files and subdirectories of the *home* directory. To change from one directory to another, the command **cd** is used.

C. Logging Off

After completing your work, it is best to log off the system. Before logging off, you should have received the prompt string "$" from the system. That is, all your commands have been completed and the system is ready for another command.

If you are using a "hard-wired" terminal, logging off is accomplished by typing an American Standard Code for Information Interchange (ASCII) End Of Text (EOT) character. On most terminals, the EOT character is

generated by holding down the "CONTROL" key and pressing the lowercase "d" key once. This is also referred to as a **CONTROL-d**. Another way to log off a hard-wired terminal is by simply turning the power to the terminal off.

For terminals connected via a phone line, the previously mentioned methods will also work, or instead, you can just hang up the phone.

Regardless of the type of terminal, the power to it should be turned off when the terminal is no longer needed. For terminals that use phone lines, depress the talk button.

D. Entering Commands

The UNIX operating system **shell** (command interpreter) serves as the interface between the user and the system. The **shell** accepts requests from the user in the form of a ***command line*** and invokes the appropriate program to fulfill the request. The **shell** prompts (i.e., notifies) the user when it is ready to accept another request. The prompt of the UNIX operating system **shell** is the primary prompt string which is by default "**$** " (a dollar sign followed by a space).

Command Line Syntax

Commands or requests to the **shell** are usually in the form of a single line, that is, a string of one or more words followed by a return. This single line request entered following the prompt is referred to as a "command line". The command line is divided into two major parts—the program name and arguments.

The first word of the command line is the name of the program to be executed. This is referred to as the **command**. All subsequent words are *arguments* to the command. Arguments are used to provide information required by the requested program.

Spaces and tabs serve as the delimiters for words on the command line. That is, all characters on the command line up to the first space or tab is the command. All characters between the first space (or tab) and the second space (or tab) is the first argument, etc. Thus, the syntax for the command line is:

command argument argument argument ... ...(RETURN)

When spaces or tabs are needed within a single argument, that argument is enclosed by ***double*** quote marks. For example, to execute a program that requires two arguments such as john l and doe. The first argument should be john and the initial l, that is, "john l" . The second argument should be doe. The required command line in this case would be:

command "john l" doe(RETURN)

Correction and Deletion

All users are likely to make mistakes, especially when typing. The UNIX operating system provides two features to correct command lines. These features are only effective for the current line (i.e., you have not ended the line with a return yet).

The first correction feature is the erase character (by default, #), and the second correction feature is the kill character (by default, @). The erase character erases the character preceding it. For example, a command line entered as

caf#t the fik#le (RETURN)

actually is "cat the file". The first # erases the first f and the second # erases the k. The erase character can be used to erase a series of characters such as in

this####the cat had kittens (RETURN)

which results in "the cat had kittens". The entire word "this" is erased by the series of # characters following it. The first # erases the s, the second # erases the i, the third # erases the h, and the fourth # erases the t. If you miscount the number of erase characters you need as in

this ###the cat had kittens (RETURN)

the result would be "ththe cat had kittens". The three erase characters erase the space, the s, and the i preceding them.

If the user needs to enter a # in the command line for some reason, preceding the # with the backslash character (\) will turn off the "erase last character" meaning of the #. For example, a command line entered as

thsi##is is the \#7# 7 cat (RETURN)

is actually "this is the # 7 cat".

The second correction feature is the kill character. The kill character deletes the entire current line. For example, the user enters the command line

command#####omma#####mmad argm##gmu##ment

when the user was trying to enter "command argument". This command line is so full of mistakes and corrections it is hard to determine if it is right. It would be best to delete the entire line and start over. The user can delete the line by ending it with an @ instead of a return. For example in this sequence

kat###catteh##he file######## the flie##e@
cat the file (RETURN)

the first line is deleted by the @ character. It is much easier to delete it and reenter it (as in the second line of the example).

If the @ character is needed in a line, the blackslash character (\) should precede it. For example, entering the line

The kill character is a \@ (RETURN)

results in "The kill character is a @".

Strange Terminal Behavior

Sometimes you can get into a state where your terminal acts strangely. For example, each letter may be typed twice (terminal may be in the half-duplex mode) or the RETURN may not cause a line feed or a return to the left margin. The user can often change this by logging out and logging back in. If logging back in fails to correct the problem, the description of the **stty**(1) command can be read to determine the appropriate action to take. To get intelligent treatment of tab characters (which are much used in the UNIX operating system) if your terminal does not have tabs, type the command

stty −tabs

and the system will convert each tab into sufficient blanks to space to the next 8-character field. If your terminal does have hardware tabs, the command **tabs**(1) will set the stops correctly for you.

Read-ahead

The UNIX operating system has full read-ahead, which means that the user can type as fast as desired, whenever the user wants, even when some command is typing at the user. If typing is done during output, the input characters will appear intermixed with the output characters, but they will be stored away and interpreted in the correct order. So the user can type several commands one after another without waiting for the first to finish or even begin.

E. Stopping a Program

Most programs can be stopped by typing the character "DEL" (perhaps called "delete" or "rubout" on your terminal). The "interrupt" or "break" key found on most terminals can also be used. In a few programs, like the text editor, "DEL" stops whatever the program is doing but leaves you in that program. Hanging up the phone will stop most programs.

F. Mail

After logging in,the user may sometimes get the following message:

You have mail.

The UNIX operating system provides a postal system so you can communicate with other users of the system. To read your mail, type the following command:

mail

Your mail will be printed, one message at a time, most recent message first. After each message, **mail**(1) waits for you to say what to do with it. The two basic responses are **d**, which deletes the message, and RETURN, which does not (so it will still be there the next time you read your mailbox). Other responses are described in the UNIX System User's Manual.

How is mail sent to someone else? Assume that "jones" is someone's login name which is recognized by **login**(1). The easiest way to send mail to "jones" is as follows:

```
mail jones
now type in the text of the letter
on as many lines as you like...
After the last line of the letter
type the character "CONTROL-d",
that is, hold down "CONTROL" and type
a letter "d".
```

The "CONTROL-d" sequence, often called End-Of-File (EOF), is used throughout the system to mark the end of input from a terminal.

For practice, send mail to yourself. (This is not as strange as it might sound—mail to oneself is a handy reminder mechanism.)

There are other ways to send mail—you can send a previously prepared letter, and you can mail to a number of people all at once. For more details, see **mail**(1).

G. Writing to Other Users

At some point, out of the blue will come a message like

Message from jones tty07...

accompanied by a startling beep. It means that Jones (jones) wants to talk to you; but unless you take explicit action, you will not be able to talk back. To respond, type the following command:

write jones

This establishes a 2-way communication path. Now whatever jones types on his terminal will appear on yours and vice versa. The path is slow, rather like talking to the moon. (If you are in the middle of something, you have to get to a state where you can type a command. Normally, whatever program you are running has to terminate or be terminated. If you are editing, you can escape temporarily from the editor—read the "Tutorial—Text Editor" section of this document.)

A protocol is needed to keep what you type from getting garbled up with what jones types. Typically it is like this:

Jones types "write smith" and waits.

Smith types "write jones" and waits.

Jones now types a message
(as many lines as necessary).
When he is ready for a reply, he
signals it by typing
(o)
which
stands for "over".

Now Smith types a reply, also
terminated by
(o).

This cycle repeats until
someone gets tired; he then
signals his intent to quit with
(oo)
for "over
and out".

To terminate
the conversation, each side must
type a "CONTROL-d" character alone
at the beginning of a line. ("DELETE" also works.)
When the other person types "CONTROL-d",
you will get the message
EOF
on your terminal.

If you write to someone who is not logged in or who does not want to be disturbed, you will be told. If the target is logged in but does not answer after a decent interval, simply type "CONTROL-d".

H. On-line Manual

The UNIX System User's Manual is kept on-line. If you get stuck on something and can not find an expert to assist you, you can print on your terminal some manual section that might help. This is also useful for getting

the most up-to-date information on a command. To print a manual section, type "man command-name". Thus to read up on the **who**(1) command, type

```
man who
```

and, of course,

```
man man
```

tells all about the **man**(1) command.

NOTES